On Boundedness Problems for Pushdown Vector Addition Systems

On Boundedness Problems for Pushdown Vector Addition Systems*

Jérôme Leroux[1], Grégoire Sutre[1], and Patrick Totzke[2]

[1] Univ. Bordeaux & CNRS, LaBRI, UMR 5800, Talence, France
[2] Department of Computer Science, University of Warwick, UK

Abstract. We study pushdown vector addition systems, which are synchronized products of pushdown automata with vector addition systems. The question of the boundedness of the reachability set for this model can be refined into two decision problems that ask if infinitely many counter values or stack configurations are reachable, respectively. Counter boundedness seems to be the more intricate problem. We show decidability in exponential time for one-dimensional systems. The proof is via a small witness property derived from an analysis of derivation trees of grammar-controlled vector addition systems.

1 Introduction

Pushdown vector addition systems are finite automata that can independently manipulate a pushdown stack and several counters. They are defined as synchronized products of vector addition systems with pushdown automata. Vector addition systems, shortly *VAS*, are a classical model for concurrent systems and are computationally equivalent to Petri nets. Formally, a k-dimensional *vector addition system* is a finite set $\boldsymbol{A} \subseteq \mathbb{Z}^k$ of vectors called *actions*. Each action $a \in \boldsymbol{A}$ induces a binary relation $\xrightarrow{\boldsymbol{a}}$ over $\mathbb{N}^k$, defined by $\boldsymbol{c} \xrightarrow{\boldsymbol{a}} \boldsymbol{d}$ if $\boldsymbol{d} = \boldsymbol{c} + \boldsymbol{a}$.

A k-dimensional *pushdown vector addition system*, shortly *PVAS*, is a tuple $(Q, \Gamma, q_{init}, \boldsymbol{c}_{init}, w_{init}, \Delta)$ where Q is a finite set of *states*, Γ is a finite *stack alphabet*, $q_{init} \in Q$ is an *initial state*, $\boldsymbol{c}_{init} \in \mathbb{N}^k$ is an *initial assignment of the counters*, $w_{init} \in \Gamma^*$ is an *initial stack content*, and $\Delta \subseteq Q \times \mathbb{Z}^k \times Op(\Gamma) \times Q$ is a finite set of *transitions* where $Op(\Gamma) \stackrel{\text{def}}{=} \{\mathsf{push}(\gamma), \mathsf{pop}(\gamma), \mathsf{nop} \mid \gamma \in \Gamma\}$ is the set of stack operations. The *size* of VAS, PVAS (and GVAS introduced later) are defined as expected with numbers encoded in binary.

Example 1.1. Consider the program on the left of Figure 1, that doubles the value of the global variable x. The $\star$ expression non-deterministically evaluates to a Boolean, as it is often the case in abstraction of programs [1]. On the right is a 1-dimensional PVAS that models this procedure: states correspond to lines in the program code, operations on the variable x are directly applied, and the call stack is reflected on the pushdown stack. □

* This work was partially supported by ANR project ReacHard (ANR-11-BS02-001).

```
1: x ← n
2: procedure DoubleX
3:     if (⋆ ∧ x > 0) then
4:         x ← (x − 1)
5:         DoubleX
6:     end if
7:     x ← (x + 2)
8: end procedure
```

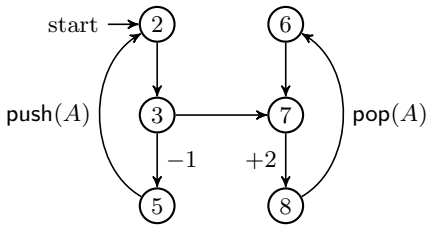

Fig. 1. A PVAS modeling a recursive program.

The semantics of PVAS is defined as follows. A *configuration* is a triple $(q, \boldsymbol{c}, w) \in Q \times \mathbb{N}^k \times \Gamma^*$ consisting of a state, a vector of natural numbers, and a stack content. The binary *step* relation $\to$ over configurations is defined by $(p, \boldsymbol{c}, u) \to (q, \boldsymbol{d}, v)$ if there is a transition $(p, op, \boldsymbol{a}, q) \in \Delta$ such that $\boldsymbol{c} \xrightarrow{\boldsymbol{a}} \boldsymbol{d}$ and one of the following conditions holds: either $op = \mathsf{push}(\gamma)$ and $v = u\gamma$, or $op = \mathsf{pop}(\gamma)$ and $u = v\gamma$, or $op = \mathsf{nop}$ and $u = v$. The reflexive and transitive closure of $\to$ is denoted by $\xrightarrow{*}$.

The *reachability set* of a PVAS is the set of configurations $(q, \boldsymbol{c}, w)$ such that $(q_{init}, \boldsymbol{c}_{init}, w_{init}) \xrightarrow{*} (q, \boldsymbol{c}, w)$. The reachability problem asks if a given configuration $(q, \boldsymbol{c}, w)$ is in the reachability set of a given PVAS. The decidability of this problem is open. Notice that for vector addition systems, even though the reachability problem is decidable [12,6], no primitive upper bound of complexity is known (see [9] for a first upper bound). However, a variant called the coverability problem is known to be ExpSpace-complete [13,11]. Adapted to PVAS, the coverability problem takes as input a PVAS and a state $q \in Q$ and asks if there exists a reachable configuration of the form $(q, \boldsymbol{c}, w)$ for some $\boldsymbol{c}$ and w. The decidability of the coverability problem for PVAS is also open. In fact, coverability and reachability are inter-reducible (in logspace) for this class [7,10]. In dimension one, we recently proved that coverability is decidable [10].

Both coverability and reachability are clearly decidable for PVAS with finite reachability sets. These PVAS are said to be *bounded*. In [8], this class is proved to be recursive, i.e. the boundedness problem for PVAS is decidable. The complexity of this problem is known to be Tower-hard [7]. The decidability is obtained by observing that if the reachability set of a PVAS is finite, its cardinality is at most hyper-Ackermannian in the size of the PVAS. Even though this bound is tight [8], the exact complexity of the boundedness problem is still open. Indeed, it is possible that there exist small certificates that witness infinite reachability sets. For instance, in the VAS case, the reachability set can be finite and Ackermannian. But when it is infinite, there exist small witnesses of this fact [13]. This yields an optimal [11] exponential-space algorithm for the VAS boundedness problem. Extending this technique to PVAS is a challenging problem.

The boundedness problem for PVAS can be refined in two different ways. In fact, the infiniteness of the reachability set may come from the stack or the counters. We say that a PVAS is *counter-bounded* if the set of vectors $\boldsymbol{c} \in \mathbb{N}^k$

such that $(q, \boldsymbol{c}, w)$ is reachable for some q and w, is finite. Symmetrically, a PVAS is called *stack-bounded* if the set of words $w \in \Gamma^*$ such that $(q, \boldsymbol{c}, w)$ is reachable for some q and $\boldsymbol{c}$, is finite. The following lemma shows that the two associated decision problems are at least as hard as the boundedness problem.

Lemma 1.2. *The boundedness problem is reducible in logarithmic space to the counter-boundedness problem and to the stack-boundedness problem (the dimension k is unchanged by the reduction).*

The stack-boundedness problem can be solved by adapting the algorithm introduced in [8] for the PVAS boundedness problem. Informally, this algorithm explores the reachability tree and stops as soon as it detects a cycle of transitions whose iteration produces infinitely many reachable configurations. If this cycle increases the stack, we can immediately conclude stack-unboundedness. Otherwise, at least one counter can be increased to an arbitrary large number. By replacing the value of this counter by ω and then resuming the computation of the tree from the new (extended) configuration, we obtain a Karp&Miller-like algorithm [5] deciding the stack-boundedness problem. We deduce the following result.

Lemma 1.3. *The stack-boundedness problem for PVAS is decidable.*

Concerning the counter-boundedness problem, adapting the algorithm introduced in [8] in a similar way seems to be more involved. Indeed, if we detect a cycle that only increases the stack, we can iterate it and represent its effect with a regular language. However, we do not know how to effectively truncate the resulting tree to obtain an algorithm deciding the counter-boundedness problem.

Contributions. In this paper we solve the counter-boundedness problem for the special case of dimension one. We show that in a grammar setting, PVAS counter-boundedness corresponds to the boundedness problem for prefix-closed, grammar-controlled vector addition systems. We show that in dimension one, this problem is decidable in exponential time. Our proof is based on the existence of small witnesses exhibiting the unboundedness property. This complexity result improves the best known upper bound for the classical boundedness problem for PVAS in dimension one. In fact, as shown by the following Example 1.4, the reachability set of a bounded 1-dimensional PVAS can be Ackermannian large. In particular, the worst-case running time of the algorithm introduced in [8] for solving the boundedness problem is at least Ackermannian even in dimension one.

Example 1.4. The Ackermann functions $A_m : \mathbb{N} \to \mathbb{N}$, for $m \in \mathbb{N}$, are defined by induction for every $n \in \mathbb{N}$ by:

$$A_m(n) \stackrel{\text{def}}{=} \begin{cases} n+1 & \text{if } m = 0 \\ A_{m-1}^{n+1}(1) & \text{if } m > 0 \end{cases}$$

These functions are *weakly computable* by the (family of) PVAS depicted in Figure 2, in the sense that:

$$A_m(n) \;=\; \max\{c \mid (\bot, n, \gamma_m) \xrightarrow{*} (\bot, c, \varepsilon)\} \tag{1}$$

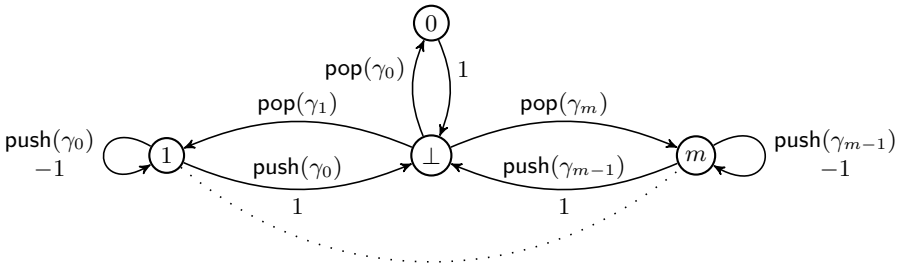

Fig. 2. One-dimensional PVAS that weakly compute Ackermann functions.

for every $m, n \in \mathbb{N}$. Indeed, an immediate induction on $k \in \{0, \ldots, m\}$ shows that $(\bot, c, \gamma_k) \xrightarrow{*} (\bot, A_k(c), \varepsilon)$ for every $c \in \mathbb{N}$. For the converse inequality, let us introduce, for each configuration $(\bot, c, w)$, the number $\theta(c, w)$ defined by

$$\theta(c, \gamma_{i_1} \cdots \gamma_{i_k}) \stackrel{\text{def}}{=} A_{i_1} \circ \cdots \circ A_{i_k}(c)$$

An immediate induction on the number of times a run come back to the state $\bot$ shows that $(\bot, c, w) \xrightarrow{*} (\bot, c', w')$ implies $\theta(c, w) \geq \theta(c', w')$. Since $\theta(c, \varepsilon) = c$, we derive that $A_m(n) \geq c$ for every c such that $(\bot, n, \gamma_m) \xrightarrow{*} (\bot, c, \varepsilon)$. This concludes the proof of Equation (1).

Notice that the reachability set of this PVAS is finite for any initial configuration. Indeed, $(\bot, c, w) \xrightarrow{*} (\bot, c', w')$ implies $\theta(c, w) \geq \theta(c', w') \geq c' + |w'|$. Therefore, there are only finitely many reachable configurations in state $\bot$. It follows that the same property holds for the other states. □

Outline. We recall some necessary notations about context-free grammars and parse trees in the next section. In Section 3, we present the model of grammar-controlled vector addition systems (GVAS) as previously introduced in [10], and reduce the counter boundedness problem for PVAS to the boundedness problem for the subclass of *prefix-closed* GVAS. We show in Section 4 that unbounded systems exhibit certificates of a certain form. Section 5 proves a technical lemma used later on and finally, in Section 6, we bound the size of minimal certificates and derive the claimed exponential-time upper bound.

2 Preliminaries

We let $\overline{\mathbb{Z}} \stackrel{\text{def}}{=} \mathbb{Z} \cup \{-\infty, +\infty\}$ denote the extended integers, and we use the standard extensions of $+$ and $\leq$ to $\overline{\mathbb{Z}}$. Recall that $(\overline{\mathbb{Z}}, \leq)$ is a complete lattice.

Words. Let A^* be the set of all finite *words* over the alphabet A. The *empty word* is denoted by ε. We write $|w|$ for the *length* of a word w in A^* and $w^k \stackrel{\text{def}}{=} ww \cdots w$ for its k-fold concatenation. The *prefix* partial order $\preceq$ over words is defined by

$u \preceq v$ if $v = uw$ for some word w. We write $u \prec v$ if u is a proper prefix of v. A *language* is a subset $L \subseteq A^*$. A language L is said to be *prefix-closed* if $u \preceq v$ and $v \in L$ implies $u \in L$.

Trees. A *tree* T is a finite, non-empty, prefix-closed subset of $\mathbb{N}^*$ satisfying the property that if tj is in T then ti in T for all $i < j$. Elements of T are called *nodes*. Its *root* is the empty word ε. An *ancestor* of a node t is a prefix $s \preceq t$. A *child* of a node t in T is a node tj in T with j in $\mathbb{N}$. A node is called a *leaf* if it has no child (i.e., $t0 \notin T$), and is said to be *internal* otherwise. The *size* of a tree T is its cardinal $|T|$, its *height* is the maximal length $|t|$ of its nodes $t \in T$. We let $\preceq_{\text{lex}}$ denote the lexicographic order on words in $\mathbb{N}^*$.

Context-free Grammars. A *context-free grammar* is a quadruple $G = (V, A, R, S)$, where V and A are disjoint finite sets of *nonterminal* and *terminal* symbols, $S \in V$ is a *start symbol*, and $R \subseteq V \times (V \cup A)^*$ is a finite set of *production rules*. We write

$$X \vdash \alpha_1 \mid \alpha_2 \mid \ldots \mid \alpha_k$$

to denote that $(X, \alpha_1), \ldots, (X, \alpha_k) \in R$. For all words $w, w' \in (V \cup A)^*$, the grammar admits a *derivation step* $w \Longrightarrow w'$ if there exist two words u, v in $(V \cup A)^*$ and a production rule (X, α) in R such that $w = uXv$ and $w' = u\alpha v$. Let $\overset{*}{\Longrightarrow}$ denote the reflexive and transitive closure of $\Longrightarrow$. The *language* of a word w in $(V \cup A)^*$ is the set $L^G_w \overset{\text{def}}{=} \{z \in A^* \mid w \overset{*}{\Longrightarrow} z\}$. The *language* of G is defined as L^G_S, and it is denoted by L^G. A nonterminal $X \in V$ is called *productive* if $L^G_X \neq \emptyset$. A context-free grammar $G = (V, A, R, S)$ is in *Chomsky normal form*[3] if, for every production rule (X, α) in R, either $(X, \alpha) = (S, \varepsilon)$ or $\alpha \in V^2 \cup A$.

Parse Trees. A *parse tree* for a context-free grammar $G = (V, A, R, S)$ is a tree T equipped with a labeling function $sym : T \to (V \cup A \cup \{\varepsilon\})$ such that the root is labeled by $sym(\varepsilon) = S$ and R contains the production rule $sym(t) \vdash sym(t0) \cdots sym(tk)$ for every internal node t with children $t0, \ldots, tk$. In addition, each leaf $t \neq \varepsilon$ with $sym(t) = \varepsilon$ is the only child of its parent. Notice that $sym(t) \in V$ for every internal node t. A parse tree is called *complete* when $sym(t) \in (A \cup \{\varepsilon\})$ for every leaf t. The *yield* of a parse tree (T, sym) is the word $sym(t_1) \cdots sym(t_\ell)$ where $t_1, \ldots, t_\ell$ are the leaves of T in lexicographic order (informally, from left to right). Observe that for every word w in $(V \cup A)^*$, it holds that $S \overset{*}{\Longrightarrow} w$ if, and only if, w is the yield of some parse tree.

3 Grammar-Controlled Vector Addition Systems

In this section we recall the notion of GVAS from [10] and show that the boundedness problem for the subclass of *prefix-closed* GVAS is inter-reducible to the counter-boundedness problem for pushdown vector addition systems.

[3] To simplify the presentation, we consider a weaker normal form than the classical one, as we allow to reuse the start symbol.

Definition 3.1 (GVAS). *A k-dimensional* grammar-controlled vector addition system *(shortly,* GVAS*) is a tuple $G = (V, \boldsymbol{A}, R, S, \boldsymbol{c}_{init})$ where $(V, \boldsymbol{A}, R, S)$ is a context-free grammar, $\boldsymbol{A} \subseteq \mathbb{Z}^k$ is a VAS, and $\boldsymbol{c}_{init} \in \mathbb{N}^k$ is an initial vector.*

The semantics of GVAS is given by extending the relations $\xrightarrow{\boldsymbol{a}}$ of ordinary VAS to words over $V \cup \boldsymbol{A}$ as follows. Define $\xrightarrow{\varepsilon}$ to be the identity on $\mathbb{N}^k$ and let $\xrightarrow{z\boldsymbol{a}} \stackrel{\text{def}}{=} \xrightarrow{\boldsymbol{a}} \circ \xrightarrow{z}$ for $z \in \boldsymbol{A}^*$ and $\boldsymbol{a} \in \boldsymbol{A}$. Finally, let $\xrightarrow{w} \stackrel{\text{def}}{=} \bigcup_{z \in L_w^G} \xrightarrow{z}$ for $w \in (V \cup \boldsymbol{A})^*$. For a word $z = \boldsymbol{a}_1 \boldsymbol{a}_2 \cdots \boldsymbol{a}_n \in \boldsymbol{A}^*$ over the terminals, we shortly write $\sum z$ for the sum $\sum_{i=1}^n a_i$. Observe that $\boldsymbol{c} \xrightarrow{z} \boldsymbol{d}$ implies $\boldsymbol{d} - \boldsymbol{c} = \sum z$.

Ultimately, we are interested in the relation $\xrightarrow{S}$, that describes the reachability relation via sequences of actions in L_S^G, i.e., those that are derivable from the starting symbol S in the underlying grammar. A vector $\boldsymbol{d} \in \mathbb{N}^k$ is called *reachable* from a vector $\boldsymbol{c} \in \mathbb{N}^k$ if $\boldsymbol{c} \xrightarrow{S} \boldsymbol{d}$. The *reachability set* of a GVAS is the set of vectors reachable from $\boldsymbol{c}_{init}$.

A GVAS is said to be *bounded* if its reachability set is finite. The associated boundedness problem for GVAS is challenging since the coverability problem for PVAS, whose decidability is still open, is logspace reducible to it. However, the various boundedness properties that we investigate on PVAS (see Section 1) consider all reachable configurations, without any acceptance condition. So they intrinsically correspond to context-free languages that are prefix-closed. It is therefore natural to consider the same restriction for GVAS. Formally, we call a GVAS $G = (V, \boldsymbol{A}, R, S, \boldsymbol{c}_{init})$ *prefix-closed* when the language L_S^G is prefix-closed. Concerning the counter-boundedness problem for PVAS, the following lemma shows that it is sufficient to consider the special case of prefix-closed GVAS.

Lemma 3.2. *The counter-boundedness problem for PVAS is logspace inter-reducible with the prefix-closed GVAS boundedness problem (the dimension k is unchanged by both reductions).*

In this paper, we focus on the counter-boundedness problem for PVAS of dimension one. We show that this problem is decidable in exponential time. The proof is by reduction, using Lemma 3.2, to the boundedness problem for prefix-closed 1-dimensional GVAS. Our main technical contribution is the following result.

Theorem 3.3. *The prefix-closed 1-dimensional GVAS boundedness problem is decidable in exponential time.*

For the remainder of the paper, we restrict our attention to the dimension one, and shortly write GVAS instead of 1-dimensional GVAS.

Example 3.4. Consider again the Ackermann functions A_m introduced in Example 1.4. These can be expressed by the GVAS with nonterminals $X_0, \ldots, X_m$ and with production rules $X_0 \vdash 1$ and $X_i \vdash -1\ X_i\ X_{i-1} \mid 1 X_{i-1}$ for $1 \leq i \leq m$. It is routinely checked that $\max\{d \mid c \xrightarrow{X_m} d\} = A_m(c)$ for all $c \in \mathbb{N}$. □

Every GVAS can be effectively normalized, in logarithmic space, by replacing terminals $a \in \mathbb{Z}$ by words over the alphabet $\{-1, 0, 1\}$ and then putting the resulting grammar into Chomsky normal form. In addition, non-productive nonterminals, and production rules in which they occur, can be removed. So in order to simplify our proofs, we consider w.l.o.g. only GVAS of this simpler form.

Assumption. We restrict our attention to GVAS $G = (V, A, R, S, c_{init})$ in Chomsky normal form and where $A = \{-1, 0, 1\}$ and every $X \in V$ is productive.

The rest of the paper is devoted to the proof of Theorem 3.3. Before delving into its technical details, we give a high-level description the proof. In the next section, we characterize unboundedness in terms of certificates, which are complete parse trees whose nodes are labeled by natural numbers (or $-\infty$). These certificates contain a growing pattern that can be pumped to produce infinitely many reachable (1-dimensional) vectors, thereby witnessing unboundedness. We then prove that certificates need not be too large. To do so, we first show in Section 5 how to bound the size of growing patterns. Then, we bound the height and labels of "minimal" certificates in Section 6. Both bounds are singly-exponential in the size of the GVAS. Thus, the existence of a certificate can be checked by an alternating Turing machine running in polynomial space. This entails the desired ExpTime upper-bound stated in Theorem 3.3.

4 Certificates of Unboundedness

Following our previous work on the GVAS coverability problem [10], we annotate parse trees in a way that is consistent with the VAS semantics. A *flow tree* for a GVAS $G = (V, A, R, S, c_{init})$ is a complete[4] parse tree (T, sym) for G equipped with two functions $in, out : T \to \mathbb{N} \cup \{-\infty\}$, assigning an *input* and an *output* value to each node, with $in(\varepsilon) = c_{init}$, and satisfying, for every node $t \in T$, the following *flow conditions*:

1. If t is internal with children $t0, \ldots, tk$, then $in(t0) \leq in(t)$, $out(t) \leq out(tk)$, and $in(t(j+1)) \leq out(tj)$ for every $j = 0, \ldots, k-1$.
2. If t is a leaf, then $out(t) \leq in(t) + a$ if $sym(t) = a \in A$, and $out(t) \leq in(t)$ if $sym(t) = \varepsilon$.

We shortly write $t : c\#d$ to mean that $(in(t), sym(t), out(t)) = (c, \#, d)$. The *size* of a flow tree is the size of its underlying parse tree. Figure 3 (left) shows a flow tree for the GVAS of Example 3.4, with start symbol X_1 and initial (1-dimensional) vector $c_{init} = 5$.

Remark 4.1. The flow conditions enforce the VAS semantics along a depth-first pre-order traversal of the complete parse tree. But, as in [10], we only require inequalities instead of equalities. This corresponds to a *lossy* VAS semantics, where

[4] Compared to [10] where flow trees are built on arbitrary parse trees, the flow trees that we consider here are always built on complete parse trees.

the counter can be non-deterministically decreased [2]. The use of inequalities in our flow conditions simplifies the presentation and allows for certificates of unboundedness with smaller input/output values. Note that equalities would be required to get certificates of reachability, but the latter problem is out of the scope of this paper.

Lemma 4.2. *For all d with $c_{init} \xrightarrow{S} d$, there exists a flow tree with $out(\varepsilon) = d$.*

Our main ingredient to prove Theorem 3.3 is a small model property. First, we show in this section that unboundedness can always be witnessed by a flow tree of a particular form, called a certificate (see Definition 4.5 and Figure 3). Then, we will provide in Theorem 6.10 exponential bounds on the height and input/output values of "minimal" certificates. This will entail the desired ExpTime upper-bound for the prefix-closed GVAS boundedness problem.

We start by bounding the size of flow trees that do not contain an iterable pattern, i.e., a nonterminal that repeats, below it, with a larger or equal input value. Formally, a flow tree (T, sym, in, out) is called *good* if it contains a node t and a proper ancestor $s \prec t$ such that $sym(s) = sym(t)$ and $in(s) \leq in(t)$. It is called *bad* otherwise. We bound the size of bad flow trees by (a) translating them into bad nested sequences, and (b) using a bound given in [8] on the length of bad nested sequences. Let us first recall some notions and results from [8]. Our presentation is deliberately simplified and limited to our setting.

Let $(S, \preceq, \|\cdot\|)$ be the normed quasi-ordered set defined by $S \stackrel{\text{def}}{=} V \times \mathbb{N}$, $(X, m) \preceq (Y, n) \stackrel{\text{def}}{\Leftrightarrow} X = Y \wedge m \leq n$, and $\|(X, m)\| = m$. A *nested sequence* is a finite sequence $(s_1, h_1), \ldots, (s_\ell, h_\ell)$ of elements in $S \times \mathbb{N}$ satisfying $h_1 = 0$ and $h_{j+1} \in h_j + \{-1, 0, 1\}$ for every index $j < \ell$ of the sequence. A nested sequence $(s_1, h_1), \ldots, (s_\ell, h_\ell)$ is called *good* if there exists $i < j$ such that $s_i \preceq s_j$ and $h_i \leq h_{i+1}, \ldots, h_j$. A *bad* nested sequence is one that is not good. A nested sequence $(s_1, h_1), \ldots, (s_\ell, h_\ell)$ is called *n-controlled*, where $n \in \mathbb{N}$, if $\|s_j\| < n + j$ for every index j of the sequence.

Theorem 4.3 ([8, Theorem VI.1]). *Let $n \in \mathbb{N}$ with $n \geq 2$. Every n-controlled bad nested sequence has length at most $F_{\omega.|V|}(n)$.*

The function $F_{\omega.|V|} : \mathbb{N} \to \mathbb{N}$ used in the theorem is part of the *fast-growing hierarchy*. Its precise definition (see, e.g., [8]) is not important for the rest of the paper. The following lemma provides a bound on the size of bad flow trees. Notice that this lemma applies to arbitrary GVAS (not necessarily prefix-closed).

Lemma 4.4. *Every bad flow tree has at most $F_{\omega.|V|}(c_{init} + 2)$ nodes.*

A good flow tree contains an iterable pattern that can be "pumped". However, the existence of such a pattern does not guarantee unboundedness. For that, we need stronger requirements on the input and output values, as defined below.

Definition 4.5 (Certificates). *A* certificate *for a given GVAS is a flow tree (T, sym, in, out) equipped with two nodes $s \prec t$ in T such that*

$$sym(s) = sym(t) \quad \text{and} \quad in(s) \leq in(t) \quad \text{and} \quad in(s) < in(t) \text{ or } out(t) < out(s)$$

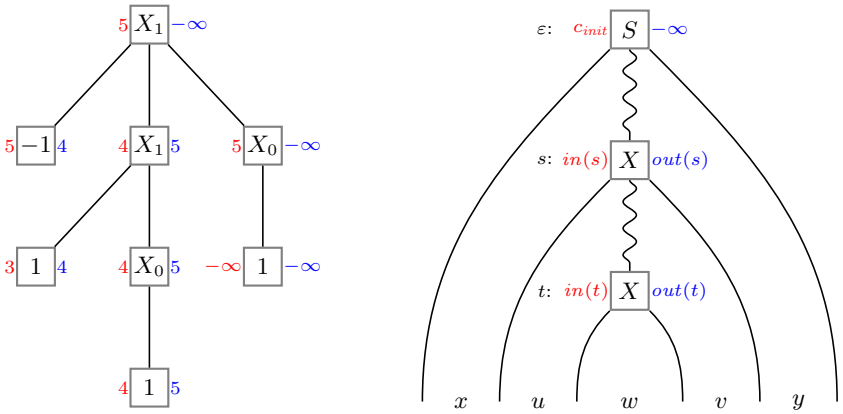

Fig. 3. Left: a flow tree for the GVAS of Example 3.4 with $c_{init} = 5$. Input and output values are indicated in red and blue, respectively. **Right:** A certificate with $sym(t) = sym(s) = X$ and yield $xuwvy \in A^*$. It must hold that either $in(s) < in(t)$ or $in(s) = in(t)$ and $out(t) < out(s)$.

We now present the main result of this section, which shows that unboundedness can always be witnessed by a certificate.

Theorem 4.6. *A prefix-closed GVAS G is unbounded if, and only if, there exists a certificate for G.*

5 Growing Patterns

Certificates depicted on Figure 3 (right) introduce words $u \in A^*$ satisfying a sign constraint $\sum u > 0$ or $\sum u = 0$. These words are derivable from words of non-terminal symbols $S_1 \dots S_k$ corresponding to the left children of the nodes between s and t. In order to obtain small certificates, in this section, we provide bounds on the minimal length of words $u' \in A^*$ that can also be derived from $S_1 \dots S_k$ and that satisfy the same sign constraint as u.

Let us first introduce the *displacement* of a GVAS G as the "best shift" achievable by a word in L^G and defined by the following equality[5]:

$$\Delta^G \stackrel{\text{def}}{=} \sup\{\textstyle\sum z \mid z \in L^G\}$$

When the displacement is finite, the following Lemma 5.1 shows that it is achievable by a complete elementary parse tree. We say that a parse tree T is *elementary* if for every $s \preceq t$ such that $sym(s) = sym(t)$, we have $s = t$. Notice that the size of an elementary parse tree is bounded by $2^{|V|+1}$.

Lemma 5.1. *Every GVAS G admits a complete elementary parse with a yield w such that $\Delta^G \in \{\sum w, +\infty\}$.*

[5] Notice that Δ^G may be negative.

Given a non-terminal symbol X, we denote by $G[X]$ the context-free grammar obtained from G by replacing the start symbol by X. We are now ready to state the main observation of this section.

Theorem 5.2. *For every sequence $S_1, \ldots, S_k$ of non-terminal symbols of a GVAS G there exists a sequence $T_1, \ldots, T_k$ of complete parse trees T_j for $G_j \stackrel{def}{=} G[S_j]$ with a yield z_j such that $|T_1| + \cdots + |T_k| \leq 3k4^{|V|+1}$, and such that $\sum z_1 \ldots z_k > 0$ if $\Delta^{G_1} + \cdots + \Delta^{G_k} > 0$, and $\sum z_1 \ldots z_k = 0$ if $\Delta^{G_1} + \cdots + \Delta^{G_k} = 0$.*

We first provide bounds on complete parse trees that witness the following properties $\Delta^G = +\infty$ and X is derivable. Formally, a nonterminal X is said to be *derivable* if there exists $w \in (A \cup V)^*$ that contains X and such that $S \stackrel{*}{\Longrightarrow} w$.

Lemma 5.3. *If $\Delta^G = +\infty$, there exists a parse tree for $G[X]$ where X is a non-terminal symbol derivable from the start symbol S with a yield uXv satisfying $u, v \in A^*$, $\sum uv > 0$, and a number of nodes bounded by $4^{|V|+1}$.*

Lemma 5.4. *For every derivable non-terminal symbol X, there exists a parse tree with a yield in A^*XA^* and a number of nodes bounded by $4^{|V|+1}$.*

Proof (of Theorem 5.2). We can assume that $k \geq 1$ since otherwise the proof is trivial. Observe that if $\Delta^{G_1} + \cdots + \Delta^{G_k} < +\infty$ then $\Delta^{G_j} < +\infty$ for every j. It follows from Lemma 5.1 that there exists a complete parse tree T_j for $G[S_j]$ with a yield w_j satisfying $\Delta^{G_j} = \sum w_j$ and a number of nodes bounded by $2^{|V|+1}$. Thus $|T_1| + \cdots + |T_k| \leq k2^{|V|+1}$ and $\sum w_1 \ldots w_k = \Delta^{G_1} + \cdots + \Delta^{G_k}$. So, in this special case the theorem is proved. Now, let us assume that $\Delta^{G_1} + \cdots + \Delta^{G_k} = +\infty$. There exists $p \in \{1, \ldots, k\}$ such that $\Delta^{G_p} = +\infty$. Lemma 5.3 shows that there exists a variable for X derivable from S_p and a parse tree T_+ for $G[X]$ with a yield uXv satisfying $u, v \in A^*$, $\sum uv > 0$, and such that $|T_+| \leq 4^{|V|+1}$. Since S_j is productive, there exists a complete elementary parse tree T_j for $G[S_j]$ with a yield $w_j \in A^*$. For the same reason, there exists a complete elementary parse tree T for $G[X]$ with a yield $w \in A^*$. As X is derivable from S, Lemma 5.4 shows that there exists a parse tree T' for G with a yield labeled by a word in $u'Xv'$ with $u', v' \in A^*$, and a number of nodes bounded by $4^{|V|+1}$. Notice that for any $n \in \mathbb{N}$, we deduce a complete parse tree T_p for $G[S_p]$ with a yield $w_p = u'u^n w v^n v'$ by inserting in T' many (n) copies of T_+ and one copy of T. Observe that $\sum w_1 \ldots w_k \geq -|w_1 \ldots w_{p-1} w_{p+1} \ldots w_k| - |u'wv'| + n \sum uv \geq -k2^{|V|+1} - 4^{|V|+1} + n$. Let us fix n to $2k4^{|V|+1} - 2$. It follows that $\sum w_1 \ldots w_p > 0$. Moreover, we have $|T_p| \leq |T| - 1 + n(|T_+| - 1) + |T'| \leq 2^{|V|+1} + n4^{|V|+1} + 4^{|V|+1} \leq (n+2)4^{|V|+1} \leq 2k4^{|V|+1}$. We derive $|T_1| + \cdots + |T_k| \leq (k-1)2^{|V|+1} + 2k4^{|V|+1} \leq 3k4^{|V|+1}$. We have proved Theorem 5.2. □

6 Small Certificates

We provide in this section exponential bounds on the height and input/output values of minimal certificates in the following sense. Let the *rank* of a flow tree

(T, sym, in, out) be the pair

$$\left(|T_{in}| + |T_{out}| \, , \sum_{t \in T_{in}} in(t) + \sum_{t \in T_{out}} out(t)) \right)$$

where $T_{in} = \{t \in T \mid in(t) > -\infty\}$ and $T_{out} = \{t \in T \mid out(t) > -\infty\}$. Notice that $T_{out} \subseteq T_{in}$. We compare ranks using the lexicographic order $\preceq_{\text{lex}}$ over $\mathbb{N}^2$ and let the rank of a certificate $(\mathcal{T}, s, t)$ be the rank of its flow tree $\mathcal{T}$.

Consider a prefix-closed GVAS $G = (V, A, R, S, c_{init})$ that is unbounded. By Theorem 4.6, there exists a certificate for G. Pick a certificate $(\mathcal{T}, s, t)$ among those of least rank. Our goal is to bound the height and input/output values of $\mathcal{T}$. Based on its assumed minimality, we observe a series of facts about our chosen certificate.

First, we observe that some input/output values in $\mathcal{T}$ must be $-\infty$, because higher values would be useless in the sense that they can be set to $-\infty$ without breaking the flow conditions nor the conditions on s and t. This observation is formalized in the two following facts.

Fact 6.1. It holds that $out(p) = -\infty$ for every proper ancestor $p \prec s$. Moreover, $in(p) = out(p) = -\infty$ for every node $p \in T$ such that $s \prec_{\text{lex}} p$ and $p \not\preceq s$.

Fact 6.2. Assume that $in(s) < in(t)$. It holds that $out(p) = -\infty$ for every ancestor $p \preceq t$. Moreover, $in(p) = out(p) = -\infty$ for all $p \in T$ with $t \prec_{\text{lex}} p$.

Next, we observe that the main branch, that contains s and t, must be short.

Fact 6.3. It holds that $|s| \leq |V|$ and $|t| \leq |s| + |V| + 1$.

The next two facts provide *relative* bounds on input and output values for nodes that are not on the main branch.

Fact 6.4. It holds that $in(p) \leq out(p) + 2^{|V|}$ for every node $p \in T$ with $p \not\preceq t$.

Fact 6.5. Let $q \in T$ and let p be the parent of q. If $p = t$ or $p \not\preceq t$, then $out(q) \leq out(p) + 2^{|V|}$. If moreover $sym(p) = sym(q)$, then $out(q) < out(p)$.

The following facts provide *absolute* bounds on the input/output values of nodes s and t. The proofs of the facts below crucially rely on Section 5. Consider the subtrees on the left and on the right of the branch from s to t. The main idea of the proofs is to replace these subtrees by small ones using Theorem 5.2.

Fact 6.6. It holds that $out(t) \leq out(s) \leq 6|V| \cdot 4^{|V|+1}$.

Fact 6.7. It holds that $in(s) \leq in(t) \leq 7|V| \cdot 4^{|V|+1}$.

Now we derive absolute bounds for the input/output values of the remaining nodes on the main branch. These are derived from Facts 6.6 and 6.7, using Facts 6.4 and 6.5 about the way in/output values propagate and the Fact 6.3 that the intermediate path between nodes s and t is short.

Fact 6.8. It holds that $out(p) \leq 4^{2(|V|+1)}$ for every ancestor $p \preceq t$.

Fact 6.9. It holds that $in(p) \leq 4^{2(|V|+1)}$ for every ancestor $\varepsilon \prec p \preceq t$.

We are now ready to derive bounds on the rank of our minimal certificate. Notice that it remains only to bound the depth and the input/output values on branches different from the main branch.

Consider therefore a node q outside the main branch, i.e., $q \not\preceq t$. Let p be the least prefix of q such that $p = t$ or $p \not\preceq t$. We first show that $out(p) \leq 4^{2(|V|+1)}$. If $p = t$ then the claim follows from Fact 6.8. Otherwise, the parent r of p satisfies $r \prec t$. Observe that the other child $\bar{p}$ of r satisfies $\bar{p} \preceq t$. The flow conditions together with the minimality of $(\mathcal{T}, s, t)$ guarantee that

- if $p = r1$ then $out(p) = out(r)$, hence, $out(p) \leq 4^{2(|V|+1)}$ by Fact 6.8, and
- if $p = r0$ then $out(p) = in(r1)$, hence, $out(p) \leq 4^{2(|V|+1)}$ by Fact 6.9.

According to Fact 6.5, the output values on the branch from p down to q may only increase when visiting a new symbol. Moreover, this increase is bounded by $2^{|V|}$. It follows that $out(r) \leq out(p) + |V|2^{|V|}$ for every node r such that $p \prec r \preceq q$. Fact 6.4 entails that $in(r) \leq out(p) + (|V| + 1)2^{|V|}$. We obtain that $\max\{in(r), out(r)\} < 4^{3(|V|+1)}$ for every node r with $p \prec r \preceq q$. Fact 6.5 also forbids the same nonterminal from appearing twice with the same output value, so $|r| \leq |p| + |V| \cdot 4^{3(|V|+1)} + 1$. Observe that $|p| \leq |t|$. We derive from Fact 6.3 that $|r| \leq 4^{4(|V|+1)}$. This concludes the proof of the following theorem.

Theorem 6.10. *A prefix-closed GVAS (V, A, R, S, c_{init}) is unbounded if, and only if, it admits a certificate with height and all input/output values bounded by $c_{init} + 4^{4(|V|+1)}$.*

Proof (of Theorem 3.3). By Theorem 6.10, a certificate for unboundedness is a flow tree of exponential height and with all input and output labels exponentially bounded. An alternating Turing machine can thus guess and verify all branches of such a flow tree, storing intermediate input/output values as well as the remaining length of a branch in polynomial space. The claim then follows from the fact that alternating polynomial space equals exponential time. □

7 Conclusion

We discussed different boundedness problems for pushdown vector addition systems [8,7], which are a known, and very expressive computational model that features nondeterminism, a pushdown stack and several counters. These systems may be equivalently interpreted, in the context of regulated rewriting [3], as vector addition systems with context-free control languages.

We observe that boundedness is reducible to both counter- and stack-boundedness. The stack boundedness problem can be shown to be decidable (with hyper-Ackermannian complexity) by adjusting the algorithm presented in [8].

Here, we single out the special case of the counter-boundedness problem for one-dimensional systems and propose an exponential-time algorithm that solves it. This also improves the best previously known Ackermannian upper bound for boundedness in dimension one.

Currently, the best lower bound for this problem is NP, which can be seen by reduction from the subset sum problem. For dimension two, PSPACE-hardness follows by reduction from the state-reachability of bounded one-counter automata with succinct counter updates [4]. For arbitrary dimensions, TOWER-hardness is known already for the boundedness problem [7,8] but the decidability of counter-boundedness for PVAS remains open.

Acknowledgments

The authors wish to thank M. Praveen for insightful discussions. We also thank the anonymous referees for their useful comments and suggestions.

References

1. Ball, T., Majumdar, R., Millstein, T.D., Rajamani, S.K.: Automatic predicate abstraction of C programs. In: PLDI. pp. 203–213 (2001)
2. Bouajjani, A., Mayr, R.: Model checking lossy vector addition systems. In: STACS. pp. 323–333 (1999)
3. Dassow, J., Pun, G., Salomaa, A.: Grammars with controlled derivations. In: Handbook of Formal Languages, pp. 101–154 (1997)
4. Fearnley, J., Jurdzinski, M.: Reachability in two-clock timed automata is pspace-complete. In: ICALP. pp. 212–223 (2013)
5. Karp, R.M., Miller, R.E.: Parallel program schemata. J. Comput. Syst. Sci. 3(2), 147–195 (1969)
6. Kosaraju, S.R.: Decidability of reachability in vector addition systems (preliminary version). In: STOC. pp. 267–281 (1982)
7. Lazic, R.: The reachability problem for vector addition systems with a stack is not elementary. CoRR abs/1310.1767 (2013)
8. Leroux, J., Praveen, M., Sutre, G.: Hyper-ackermannian bounds for pushdown vector addition systems. In: CSL/LICS (2014)
9. Leroux, J., Schmitz, S.: Demystifying reachability in vector addition systems. In: LICS (2015)
10. Leroux, J., Sutre, G., Totzke, P.: On the coverability problem for pushdown vector addition systems in one dimension. In: ICALP. pp. 324–336 (2015)
11. Lipton, R.J.: The reachability problem requires exponential space. Tech. Rep. 63, Yale University (Jan 1976)
12. Mayr, E.W.: An algorithm for the general Petri net reachability problem. In: STOC. pp. 238–246 (1981)
13. Rackoff, C.: The covering and boundedness problems for vector addition systems. TCS 6(2), 223–231 (1978)